جلسه پنجم پروژه UTAXI - پیاده‌سازی کد با زبان C++

جلسه پنجم پروژه **UTAXI** با تمرکز کامل بر شروع فرآیند پیاده‌سازی کد به زبان **C++** برگزار شد. در این جلسه تمامی اعضای تیم مشارکت فعالی داشتند تا پایه‌های کد پروژه ایجاد شود و ساختارهای اصلی سیستم تعریف شوند. این جلسه به دلیل اهمیت بالای پیاده‌سازی فنی و تعیین مسیر کدنویسی، یکی از نقاط کلیدی پروژه به‌شمار می‌رفت.

---

1. مقدمه جلسه

جلسه با مرور مختصری از پیشرفت پروژه در جلسات قبلی آغاز شد. **امین یوسفی**، مدیر پروژه، اهداف این جلسه را تشریح کرد و تأکید کرد که تیم باید:

- ساختارهای پایه سیستم را در زبان C++ ایجاد کند.
- استانداردهای کدنویسی مشترک را تعیین کند.
- پیاده‌سازی اولیه کلاس‌های اصلی را آغاز کند.
- اصول همکاری در GitHub را رعایت کرده و تغییرات را به درستی مدیریت کنند.

لینک‌های مرتبط با پروژه برای دسترسی اعضای تیم یادآوری شد:

- لینک اصلی پروژه
- کدهای منبع

---

2. اهداف جلسه

اهداف اصلی جلسه پنجم شامل موارد زیر بود:

1. تعریف و پیاده‌سازی **کلاس‌های اصلی پروژه**:
   - کلاس‌های اصلی مانند **کاربر (User)**، **راننده (Driver)**، **مسافر (Passenger)**، **درخواست سفر (RideRequest)** و **مدیر سیستم (SystemManager)** مشخص شدند.
2. ایجاد و تعیین **استانداردهای کدنویسی** تیم.
3. شناسایی وابستگی‌ها و روابط بین کلاس‌ها.

4. شروع فرآیند پیاده‌سازی اولیه کد و مستندسازی.
5. آموزش نکات همکاری و مدیریت تغییرات در **GitHub**.

---

3. تقسیم وظایف میان اعضای تیم

بر اساس تخصص و توانایی‌های هر یک از اعضا، وظایف به صورت زیر تقسیم شد:

- **مبینا مهرآذر**: طراحی و پیاده‌سازی کلاس **DatabaseManager** برای مدیریت ذخیره‌سازی اطلاعات کاربران و سفرها.
- **محمدرضا نعمتی**: توسعه کلاس **BackendController** برای مدیریت APIها و ارتباطات سرور.
- **آرین باستانی**: طراحی رابط کاربری کنسول (CLI) و ایجاد تعاملات اولیه بین کاربران و سیستم.
- **امین یوسفی**: نظارت بر روند پیاده‌سازی، اطمینان از هماهنگی بین اعضای تیم، و ادغام نهایی کدها.
- **محمد امانلو**: پیاده‌سازی **مدیریت خطا**، ایجاد تست‌های اولیه، و تضمین کیفیت کدهای تولید شده.

---

4. استانداردهای کدنویسی

تیم تصمیم گرفت از استانداردهای زیر در کدنویسی استفاده کند:

- **نام‌گذاری متغیرها و توابع** به صورت **CamelCase**.
- هر کلاس در یک فایل **هدر (.h)** تعریف شود و پیاده‌سازی آن در یک فایل **منبع (.cpp)** قرار گیرد.
- استفاده از **الگوهای طراحی شی‌گرا** (Object-Oriented Design) برای حفظ انسجام کد.
- مستندسازی تمام کلاس‌ها و توابع با کامنت‌های واضح به زبان انگلیسی.
- اجرای تست‌های محلی پیش از ثبت تغییرات در GitHub.

---

5. فرآیند پیاده‌سازی

اعضای تیم کار خود را روی کلاس‌های مشخص شده آغاز کردند. جزئیات کدنویسی هر بخش به شرح زیر است:

5.1. کلاس User (کاربر)

- طراحی به عنوان کلاس پایه برای رانندگان و مسافران.

- ویژگی‌ها:
  - `string name;` (نام)
  - `string email;` (ایمیل)
  - `string phoneNumber;` (شماره تماس)
  - `int userID;` (شناسه کاربر)
- توابع:
  - `void registerUser();` (ثبت‌نام کاربر)
  - `void updateUserDetails();` (به‌روزرسانی اطلاعات کاربر)

5.2. کلاس Driver (راننده)

- این کلاس از کلاس User ارث می‌برد و ویژگی‌های خاص رانندگان را اضافه می‌کند.
- ویژگی‌ها:
  - `string vehicleDetails;` (جزئیات خودرو)
  - `int driverRating;` (امتیاز راننده)
- توابع:
  - `void acceptRide();` (پذیرش سفر)
  - `void updateDriverAvailability();` (به‌روزرسانی وضعیت راننده)

5.3. کلاس Passenger (مسافر)

- این کلاس نیز از User ارث می‌برد و ویژگی‌های خاص مسافران را تعریف می‌کند.
- توابع:
  - `void requestRide();` (درخواست سفر)
  - `void rateDriver();` (امتیازدهی به راننده)

5.4. کلاس RideRequest (درخواست سفر)

- مدیریت درخواست‌های سفر را برعهده دارد.
- ویژگی‌ها:
  - `int rideID;` (شناسه سفر)
  - `string pickupLocation;` (مبدأ)
  - `string dropoffLocation;` (مقصد)
  - `int passengerID;` (شناسه مسافر)
  - `int driverID;` (شناسه راننده)

- توابع:
  - `void createRide();` (ایجاد درخواست سفر)
  - `void cancelRide();` (لغو درخواست سفر)

5.5. کلاس SystemManager (مدیر سیستم)

- وظیفه مدیریت کل سیستم و تعامل بین کلاس‌های دیگر را برعهده دارد.
- توابع:
  - `void initializeSystem();` (راه‌اندازی سیستم)
  - `void handleRideRequests();` (مدیریت درخواست‌های سفر)

---

6. چالش‌ها و راه‌حل‌ها

در حین پیاده‌سازی، چالش‌های زیر پیش آمد و تیم با همکاری آن‌ها را رفع کرد:

1. **هماهنگی در GitHub**: برخی اعضا در مدیریت نسخه‌ها و حل تعارضات در GitHub مشکل داشتند.
   آموزش کوتاهی درباره استفاده از Branchها و Merge کردن تغییرات ارائه شد.
2. **مدیریت وابستگی‌ها**: نیاز به تعریف دقیق روابط بین کلاس‌ها وجود داشت تا از وابستگی‌های چرخشی جلوگیری شود.
3. **رفع خطاهای کامپایل**: خطاهای اولیه در پیاده‌سازی کدها با بررسی دقیق و اصلاحات برطرف شد.

---

7. برنامه‌ریزی جلسات آینده

- تکمیل پیاده‌سازی کلاس‌ها و افزودن ویژگی‌های پیشرفته.
- نوشتن تست‌های جامع برای اطمینان از عملکرد صحیح سیستم.
- ادغام نهایی کدها و اجرای تست‌های جامع سیستم.
- برنامه‌ریزی برای پیاده‌سازی قابلیت‌های پیشرفته مانند:
  - **سیستم امتیازدهی رانندگان و مسافران.**
  - **محاسبه هزینه سفر.**
  - **مدیریت پایگاه داده.**

---

8. نتیجه‌گیری جلسه

این جلسه با تمرکز بر پیاده‌سازی فنی به زبان C++، تنظیم ساختارهای پایه، و هماهنگی بین اعضا پیشرفت قابل‌توجهی داشت. در پایان:

- ساختارهای اولیه کلاس‌ها تکمیل شدند.
- استانداردهای کدنویسی مشخص و اعمال شدند.
- همکاری تیمی در GitHub بهبود یافت.